7/9/4   (Item 1 from file: 16)
DIALOG(R)File 16:Gale Group PROMT(R)
(c) 2004 The Gale Group. All rts. reserv.

03318380   Supplier Number: 44587050 (THIS IS THE FULLTEXT)

FPGAs integrate datacom's paths

Electronic Engineering Times, pS24
April 11, 1994
ISSN: 0192-1541
Language: English   Record Type: Fulltext
Document Type: Magazine/Journal; Trade
Word Count: 1901

TEXT:

BY JOHN TSIMARAS
MEMBER OF TECHNICAL STAFF
AT&T ATM PLATFORM DIVISION
RED BANK, N.J.
AND ALAN CUNNINGHAM
MEMBER OF TECHNICAL STAFF
AT&T FPGA APPLICATIONS
ALLENTOWN, PA.

Traditional datacom applications have largely relied on discrete logic components for such supporting circuits as data registers; first in, first outs (FIFOs); last in, first outs (LIFOs); control; and data paths. Advanced field-programmable gate array (FPGA) architectures, however, offer the datacom system designer on-chip SRAM functions plus control logic to allow cost-effective integration of the necessary functions. Equally important, circuit designs can be custom tailored to meet a datacom application's special requirements.

FPGAs with on-chip SRAM and supporting logic are especially valuable because they hand datacom-system designers the necessary integrated silicon to implement the storing, checking and control functions used for processing small packets of data. Because all the required functions can be performed on-chip, higher system frequencies can be attained at a lower power consumption.

Unfortunately, to create very wide RAM functions, the designer must use combinatorial logic to link the SRAM blocks. As the width of the RAM increases, the combinatorial logic eventually becomes so wide that it makes more sense to use discrete RAM. One vendor uses the 'rule of thumb' that a RAM with a depth greater than 256 words should be implemented in discrete RAM. Such a rule should be used only as a guide. The final decision should be based on a number of constraints, including board space, power consumption and system timing.

Off-the-shelf discrete ICs such as FIFOs are also available for datacom designs. However, those standard parts may not have the required timing or the necessary control. By using an FPGA, the designer can customize the function to meet the system's timing requirements, allow special controls and access system-specific flags. The designer can also pare power and pc-board real estate requirements.

Chips available on the market with on-chip SRAM include the Xilinx Inc. XC4000, Intel Corp. Flexlogic and AT&T Orca. The accompanying table shows each device's SRAM-block organization.

In the XC4000, each logic block operates as a complex logic function, a 32-word X 1-bit block of SRAM or two 16 X 1 SRAM blocks.

Each logic block in Flexlogic FPGAs is called a configurable function block (CFB) and can be configured as a 24V10 logic block or a 128 X 10 SRAM block. The SRAM is accessed in a conventional way by using 7 bits of the 24 signal fan-in as address data and 10 bits as data-in. As in the XC4000, different -sized SRAMs are possible by cascading multiple CFBs to increase the memory depth or width; in the Intel device, however, one must add RAM in 10-bit wide chunks.

Orca FPGAs are also configurable, but their configurability is based on a lookup table (LUT) architecture. LUTs can be configured as a complex logic function, SRAM or ROM. One logic element, or programmable function unit (PFU), can be used to implement one 16 X 4, two 16 X 2, or one 16 X 2 SRAM block, with the remaining one half of the PFU used for random-logic consolidation.

In memory mode, each half-LUT (HLUT) is configured independently so that logic designers can opt for the 16 X 2 in one HLUT, a 16 X 4 using both HLUTs, or the 16 X 2 in one HLUT and a logic function of five input variables or less in the other HLUT. Two or more programmable logic cells (PLCs) in Orca are used to increase memory depth to a value of greater than 16. Deeper memories are more complex and require additional logic. To increase the memory depth, data outputs from each PLC are multiplexed. Combinatorial logic is required to access the write enables of the individual PLCs. An example of a 32 X 4 RAM block would be created by using two 16 X 4 blocks in the Orca FPGA. Similar techniques can be used for the Xilinx XC4000 and the Intel Flexlogic.

The 32 X 4 RAM module would require two Orca PLCs in the 16 X 4 mode. The two blocks would share the same address and data lines.

The Orca FPGA also has an advantage because within each PLC are four tri -stateable buffers, called bidis, that can be used to create a 2:1 multiplexer. The outputs of the bidis would be tied together to create the 4 -bit output bus.

Two or more PLCs are used again to increase a SRAM's word size, such as 16 X 8. The PLC's address and write enable are tied together, and data is different for each PLC. The address locations and word size are increased by using a combination of the two techniques.

On-chip SRAM-based FIFOs are particularly useful in network-system designs, whether the network type is Asynchronous Transfer Mode (ATM), Fiber Channel, token ring, Fiber Distributed Data Interface (FDDI) or Ethernet.

FIFOs synchronize data between two asynchronous systems and usually need dual -ported SRAM. In that case, separate read and write address lines simultaneously access the SRAM. On-chip SRAM in today's FPGAs is single ported, however, so that only one set of address inputs goes to the SRAM. But converting a single-port SRAM into a dual-port type is easy. Read and write address lines are time-division multiplexed to form a single address input. The read count has the current read address, and the write count points to the current write location. Both counter outputs are multiplexed together to form the single address SRAM input.

A read accesses the SRAM and is then followed by a write, and so on. The FIFO processes each request in two clock cycles. The control block arbitrates between read and write. When a read and a write request occur simultaneously, the arbiter performs either a read before a write (if the FIFO isn't empty) or a write before a read (if the FIFO is empty). A comparator circuit checks the read and write count pointer outputs to detect either full or empty conditions for the FIFO. Control circuitry doesn't acknowledge a read request from an empty FIFO or a write request from a full FIFO. Though it is synchronous internally, the FIFO supports asynchronous handshakes to the request and data ports.

The accompanying figure shows a more datacom-specific on-chip SRAM FIFO, used to hold an ATM data packet or cell. An ATM cell is constructed of 53 bytes. The first 5 bytes contain the header; the remaining 48 bytes hold the information field. At the user-network interface, the header is broken into the Generic Flow Control (GFC), the Virtual Path Identifier (VPI), the Virtual Channel Identifier (VCI), the Payload Type Identifier (PTI), the Cell Lost Priority (CLP) and the Header Error Control (HEC).

The first 5 bytes are **broken** up as follows: the GFC uses the 4 **most significant** bits of the first byte; the VPI uses the 4 **least significant** bits of the first byte and the 4 **most significant** bits of the second byte; the VCI uses the 4 **least significant** bits of the second byte, the entire third byte and the 4 most significant bits of the fourth byte; the PTI uses the next 3 bits of the fourth byte; the CLP uses the least significant bit of the fourth byte; and the HEC uses the entire fifth byte.

The FIFO design in the ATM application holds the 53-byte cell and performs a cycle redundancy check (CRC) on the 5-byte header. If the header is found to be error free, the complete cell is forwarded to system circuitry. But if an error is detected when the 5-byte header is verified, the complete 53-byte cell is immediately discarded. That's done when the write counter is cleared.

Based on a 5,000-gate ATT1C05 device, the 64-byte FIFO uses a total of 18 PLCs. The read and write counters each use two PLCs. Eight more are used for storage, one for the two-way control arbiter, two for the 2:1 multiplexer, and three for data validation.

The purpose of the single-cell FIFO implemented on-chip in the ATM application is to validate and discard any 'corrupt' ATM cells before they are allowed to enter the network. The on-chip FIFO is used during that validation and is not intended to serve any large buffering or rate -decoupling purpose. Those functions are instead left to external discrete FIFO devices. The intent of the application is to screen out corrupt cells and pass only valid cells to the external FIFO(s) that may serve as system buffers.

Another application for on-chip RAM is as a register file. A register file can be thought of a set of registers that share a common multiplexed output. To be able to use on-chip RAM for register files, the outputs from different registers cannot be accessed simultaneously. Before on-chip RAM was available, designers would create a register file using individual flip-flops - an approach that clearly requires an inordinate amount of resources.

Some data networking products use proprietary system addressing schemes to move data from one user port to another within the unit. Such addressing schemes are likely to be tailored to the specific hardware architecture of the unit. Hence, data conforming to certain standards must be translated to meet those addressing schemes so it can be used within the system. Likewise, when the data leaves the system, it must be translated back to meet the original standard. Register files can be used to translate the data.

This application explores one possible (and simple) translation, wherein the 4 most significant bits of the VPI and the entire GFC are replaced by a byte of routing information that is extracted from the on-chip RAM. The VPI is thus effectively expanded from 8 bits to 12. The expansion is beneficial and, in most cases, necessary. If one user port (or end point) is allowed 256 possible VPIs, then a switch capable of supporting multiple end points should be able to identify and support a larger set of connections. In the example cited here, the additionally expanded 4 bits could contain a destination port number.

Significant bits

In the register file application, the translator uses the 4 most significant bits of the VPI as the address into a 16 X 8 on-chip RAM. Those access a byte of data that, when used in conjunction with the 4 least significant bits of the VPI, create a 12-bit VPI.

The circuit comprises four 2:1 multiplexers, a small control block, an 8-bit register and a 16 X 8 RAM. When a cell of information is transmitted, the control block uses 4 bits of the header (the 4 most significant bits of the VPI) as the address to the RAM.

That address accesses 8 data bits, which are then used to replace the first byte of the cell. All 52 remaining cells are passed through an 8-bit register and are not altered. Inversely, when the cell leaves the switch, the data must be transformed back to its original state.

The above application requires seven PLCs within the Orca device. A similar implementation using discrete logic clearly would not fully utilize the off -chip RAM and would require a large amount of board real estate. If the design was implemented in an FPGA that did not have the added advantage of on -chip RAM, the register file would require 128 flip-flops or 32 PLCs within the Orca family of FPGAs.